How can programming languages create more interactive VR experiences?

Virtual reality (VR) is a technology that immerses users in simulated environments, where they can interact with digital objects and characters. Programming languages are the tools that enable developers to create these VR experiences, by defining the logic, graphics, audio, and input of the VR applications. But how can programming languages create more interactive VR experiences.

We will explore some of the features and challenges of programming languages for VR, and how they can enhance the realism, immersion, and engagement of VR users.

1. VR Programming Languages

There are many programming languages that can be used for VR development, depending on the platform, engine, and framework of choice. Some of the most popular ones are C#, C++, Java, JavaScript, and Python. These languages offer different advantages and disadvantages, such as performance, flexibility, simplicity, and compatibility. For example, C# is widely used with Unity, a popular game engine for VR, because it is easy to learn, has a large community, and supports cross-platform development. C++, on the other hand, is preferred by some developers who use Unreal Engine, another game engine for VR, because it offers more control, speed, and optimization.

2. VR Scripting Languages

Scripting languages are a type of programming languages that are interpreted at runtime, rather than compiled beforehand. They are often used to create dynamic and interactive features in VR applications, such as animations, events, behaviors, and user interfaces. Scripting languages are usually easier to write and modify than compiled languages, and can be embedded in other languages or engines. Some examples of scripting languages for VR are Lua, Python, and JavaScript. Lua is a lightweight and fast scripting language that can be integrated with C or C++, and is used by some VR frameworks, such as LÖVR and Godot. Python is a versatile and powerful scripting language that can be used with various VR libraries, such as PyOpenVR and PyOculus. JavaScript is a widely used scripting language that can be used to create web-based VR applications, using frameworks such as A-Frame and Three.js.

3. VR Domain-Specific Languages

Domain-specific languages (DSLs) are programming languages that are designed for a specific domain or problem, rather than for general purposes. They are often used to simplify and abstract complex or low-level tasks, and to increase the productivity and readability of the code. DSLs can be used to create more interactive VR experiences, by allowing developers to focus on the high-level logic and design of the VR applications, rather than on the technical details and implementation. Some examples of DSLs for VR are VRML, X3D, and A-Frame. VRML and X3D are DSLs that describe 3D scenes and objects in VR, using a declarative syntax that resembles HTML. A-Frame is a DSL that builds on HTML and JavaScript, and enables developers to

create web-based VR applications using custom HTML tags and attributes.

4. VR Programming Challenges

Programming languages for VR face some challenges and limitations that can affect the quality and interactivity of the VR experiences. These include performance, input, and testing. To ensure a smooth and realistic experience, programming languages for VR must be optimized and efficient, avoiding unnecessary computations and memory allocations. Additionally, they must be able to handle and process various inputs from controllers, headsets, trackers, gestures, and voice. Lastly, programming languages for VR need to provide tools and frameworks that facilitate the testing and debugging process in the VR environment, allowing developers to monitor and modify the code in real time.

5. VR Programming Opportunities

Programming languages for VR offer some opportunities and possibilities that can enhance the interactivity and creativity of the VR experiences. For example, AI can be used to create more intelligent and adaptive VR applications that respond to user actions, preferences, and emotions. Leveraging AI techniques such as machine learning, natural language processing, and computer vision can create a more immersive and engaging experience. Multiplayer capabilities enable users to connect with other people and environments. Networking, synchronization, communication, and authentication features provide more interactive and fun experiences. Additionally, mixed reality (MR) technologies like augmented reality (AR), haptic

feedback, and spatial audio can create a more realistic and immersive VR experience.

Getting Started with VR Development in 2024

Where do you get started if you want to make the next great Mixed/Augmented/Virtual Reality experience?

In 2024, VR development offers the kind of new opportunity most coders might consider “once in a decade” if not once in a lifetime.

Platforms in VR Development

To start wrapping your head around concepts in XR, especially if you’re a student coming to XR, you might want to do some general reading on introductory topics. For instance, there are different platforms for making experiences:

- **PCVR** — PCVR encompasses the broad notion of Windows-based video games that run on VR headsets, many of which are exclusive and well worth playing such as Half-Life: Alyx, Skyrim VR, to a lesser extent (but totally en vogue due to the TV series) Fallout 4 VR. PCVR offers the most raw capabilities for a platform, but keep in mind, one needs a very powerful computer to run PCVR games and the hardware itself

- **Meta Quest** — The Quest 3 is definitely the most popular standalone headset for getting started content on the web right now and my personally recommended device for anyone who has yet to buy any hardware. The Quest 2 is now an ultra-affordable hardware option for aspiring developers who don't want to make the plunge, but remember, it can't do mixed/augmented reality. The headsets themselves can also connect to a computer to do PCVR development and has a WebXR-enabled browser out-of-the-box, so it's a very flexible device for development.
- **Apple Vision Pro** — The newly released Apple Vision Pro (AVP) is, in my perception, a software development kit for aspiring developers rather than a consumer product at the moment. It's so much more capable than anything else, but so much more expensive. If you are reading this and can code, you're probably the only viable market beyond the earliest of early adopters out there. It cannot connect to a PC, but can run iOS and Vision OS apps — plus has nascent game engine support in Unity. Additionally, with some power user options turned on, its browser can run WebXR experiences.

PSVR2 is a great balance of affordability and performance compared as far as playing VR games. However, the fact one needs a Playstation Development Kit to get started means it's likely a not starter for enthusiasts getting into coding since it's the cost of the AVP to get going and a much smaller audience compared to other platforms.

VR Software Development Toolsets

For instance, the notion that you can take many different paths to making a game:

- **Native Development** — Such as making an Android in VR native app on Quest or a VisionOS app for Apple Vision Pro (AVP). These seem most suited to non-gaming, productivity-oriented application. AVP developers who want in on the return of people actually paying for apps for Apple devices thanks to the AVP or prepare for a future where some of the B2B software economy will be built in XR, one will likely want to try native development.
- **Game Engine** — Development using a ready-made toolset using popular professional engines like Unity, Unreal, or the open source Godot. Each of those has XR features built-in plus strong community support both from the creators and online tutorials from the ecosystem at large — though Unity seems like the most robust starting place for beginners and Unity has support for AVP thanks to a partnership with Apple. This is definitely the best path for someone serious about making media and entertainment products and delivering them to a real app store.
- **WebXR Development** — If you want ultimate cross-platform support with lightning fast development time, you might try using the budding WebXR standard and the ecosystem on top of it to make the browsers pre-loaded into all of these devices to make your first XR experience. A framework like Babylon.js or web game

engines like Wonderland Engine can get you started quickly and virtually any VR device can be used for live testing. The capabilities of WebXR are less robust, but fast iteration and broad reach may outweigh raw features for your idea. This is an especially good idea if you just want to understand VR design and have a nice proof-of-concept demo for your portfolio without necessarily wanting to make a "real product" out of it.

For a video summary of some of these points of consideration, check out this video on YouTube from Valem Tutorials who has more learning content, later we shall look at Unity development on Meta Quest. It runs over the platforms and options, with some gentle introduction to terms throughout the video to start you on the basics:

https://youtu.be/i3DbJwy0R6E

From here, let's get into individual devices and how to approach building your first apps for them. Given the options above, one has at least nine combinations, but truly there are only a few likely paths worth true consideration.

Development Hardware

One does need a personal computer of some kind (Mac, PC, Linux) to do VR development. Most recent machines can do it, though it seems like laptops need discrete graphics cards. One can install the Oculus Desktop App and it will check if your machine can run the OculusXR runtime. If it supports it, you can likely do VR development on that

machine. If it doesn't support it, you might still be able to do WebXR development on the machine.

The Default Path: Why Meta Quest?

The popular narrative about Meta's entrance into the VR market is definitely that Mark Zuckerberg has set billions of dollars on fire to build the Metaverse, but not yet with any fruits for Meta to show for it..

The Meta Quest is not only the arguable champion of price-to-features in the VR market right now, but it has the most dynamic software ecosystem. There is an estimated 20 million Quest 2 headsets sold at time of publishing and at least 1 million Quest 3 headsets sold thus far. More importantly, the collective Meta Quest software ecosystem is reported to have passed $2 billion cumulative sales; the kind of ecosystem Steam might have had back in 2008 or 2009 based on estimates, back when it was niche but about to go mainstream.

While that may seem small, the most important aspect is the growth of the platform. Well, Meta is making the same opportunity for anyone willing to sit down and learn today.

The Default Path: Getting Started on Meta Quest with Unity

The largest ecosystem at the intersection of hardware and software development kit (SDK) in VR at the moment appears to be Unity development on the Meta Quest, thus if you want a "default path"

Quick links to get started would be:

1. Research Unity VR Capabilities a bit to understand the capabilities
2. Download Unity onto Your Windows, Mac, or Linux machine
3. Bookmark and start reading the official Unity VR Development Docs
4. Find tutorials, articles or videos, online to follow for getting started content, **paying special attention to when they were made to avoid potentially learning old techniques**

In the Unity store, you'll find the modern Meta XR All-In-One SDK that is the new starting point for Unity apps targeting Oculus apps for VisionOS 65 or later. Meta also offers the Building Blocks for starting points for key elements of VR/AR games in Unity.

https://youtu.be/FwC81qCi-Oc

Once you have tried some tutorials, the best way to find more advanced examples is open source projects from which you can come to understand more complex examples of experiences.

- Oculus Samples repo on GitHub which has native, Unreal, and Unity examples,

- Valem Tutorials list of best open source VR unity projects
- Open source apps on the SideQuest Oculus Game directory where you can find unpublished Quest apps, many of which have open source projects you can find as links from their SideQuest entries or by searching the web for the author or game by name

PCVR with Unity

If you would like to focus on the more general PCVR rather than Meta Quest specifically — you can just plug a Quest headset (or HTC Vive, Valve Index, etc) into a PC. You might try starting with the Valem Tutorial's video on "Learning VR Development in 3 Hours — Unity VR Tutorial Complete Course", which uses a slightly different configuration of packages in Unity to make a more general-purpose VR experience.

https://youtu.be/YBQ_ps6e71k

PCVR with Godot

If you want to make Richard Stallman proud and, more importantly, ensure you don't rely on a large company that can change its fee structure at any time to ruin the business potential for your app or build on a platform by a tech giant known for privacy concerns, then making a PCVR app using Godot Engine would be the ideal starting point. The Build a VR Game in Godot video series seems like the most comprehensive starting point.

https://youtu.be/fxZoXfX4oBo

Apple Vision Pro with VisionOS

If you want to try building an Apple Vision Pro app, I would recommend :

- If you want to make AR apps for AVP, then check out the Apple VisionOS documentation then check out some VisionOS app tutorials online. Again, make sure you have an Apple MacBook running Apple Silicon (the M1, M2, etc series processors).
- If you want to make immersive experiences for Apple Vision Pro, check out the Unity PolySpatial features, which enables 3D experiences on Apple Vision Pro. As of this writing, it seems they have made it a pro feature, so you have only have 30-days of free trial to dig in if you don't want to start paying.

Conclusion

XR as an important future media.